On scheduling malleable jobs to minimise the total weighted completion time

Ruslan Sadykov [*,**]

[*] *INRIA Bordeaux — Sud-Ouest, France*
[**] *Institut de Mathématique de Bordeaux, 351 cours de la Libération, 33405 Talence, France*

Abstract: This paper is about scheduling parallel jobs, i.e. which can be executed on more than one processor at the same time. Malleable jobs is a special class of parallel jobs. The number of processors a malleable job is executed on may change during the execution.
In this work, we consider the NP-hard problem of scheduling malleable jobs to minimize the total weighted completion time or mean weighted flow time. For this problem, we introduce an important dominance rule which can be used to reduce the search space while searching for an optimal solution.

Keywords: Combinatorial mathematics; scheduling algorithms; total completion time, parallel jobs; malleable jobs.

1. INTRODUCTION

With the emergence of new production, communication and parallel computing system, the usual scheduling requirement that a job is executed only on one processor has become, in many cases, obsolete and unfounded. Therefore, parallel jobs scheduling is becoming more and more widespread.

The malleable jobs which have been introduced by Turek et al. (1992) is a special class of the parallel jobs. The number of processors a malleable job is executed on may change during the execution. For an overview of malleable jobs scheduling, we refer to Dutot et al. (2004) and Ludwig (1995), for an application in parallel optimization, to Blayo et al. (2000), and for an application in textile industry to Serafini (1996).

We now define the problem. A set $N = \{1, \dots, n\}$ of preemptive jobs should be processed on a set $M = \{1, \dots, m\}$ of identical machines. Each job $j \in N$ has a processing time (or a surface) p_j, a maximum number δ_j of machines the job can be processed on and a weight w_j. We assume that jobs are *work preserving*, i.e. the execution of a job j on q machines at the same time reduces its processing time by q: $p_j(q) = p_j/q$. We denote by $C_j(\pi)$ the completion time of job j in schedule π. The objective is to find a schedule π which minimizes the total weighted completion time

$$F(\pi) = \sum_{j \in N} w_j C_j(\pi). \tag{1}$$

Note that the function (1) is equivalent to mean weighted flow time, as all the jobs available for processing from time 0. In the standard scheduling notation, the problem is denoted as $P \mid var,\ p_j(q) = p_j/q,\ \delta_j \mid \sum w_j C_j$, see Drozdowski (2004).

The problem we consider is a generalisation of the problem to schedule preemptive non-parallel jobs ($\delta_j = 1,\ \forall j \in N$) on identical parallel machines to minimise the total weighted completion time $P \mid pmtn \mid \sum w_j C_j$. As the latter is NP-hard in the strong sense, as shown by Garey and Johnson (1979), our problem is also NP-hard.

The contribution of this paper is an important dominance rule for the problem. We will show that there exists an optimal schedule in which, for every job, the number of processors this job is executed on do not decrease over time while the job is being processed. We say that such schedules satisfy *ascending property*. This result allows us to reduce the search for an optimal schedule of the problem to this class of dominant schedules.

2. DEFINITIONS

A *piece* of a job j is a non-preemptive part of j processed on some machine i. Note that, by the work preserving assumption, the sum of the lengths of all the pieces of a job j in a schedule should be equal to the surface p_j of this job. The completion time of a piece u of a job j in a schedule π will be denoted as $C_j^u(\pi)$. Remember that the completion time $C_j(\pi)$ of a job j in a schedule π is equal to the maximum completion time for all pieces of job j.

A piece of job in a schedule is *early* if its completion time is strictly smaller than the completion time of the job it belongs to.

A schedule satisfies the *ascending property* if it does not contain early pieces.

3. THE MAIN RESULT

Theorem 1. There exists an optimal schedule which satisfies the ascending property.

Proof. To prove the theorem, we will show that it is possible to transform an optimal schedule which does not satisfy the ascending property into another optimal schedule in which

- either the total number of pieces of jobs is strictly decreased,
- or the number of early pieces is strictly decreased.

Applying this transformation to an optimal schedule a finite number of times, we can obtain an optimal schedule without early pieces.

Let π be an optimal schedule of the problem, and π does not satisfy the ascending property. Therefore, there is an early piece of a job in π. Let piece q of job a be an early piece with maximum completion time in π. Also, let $K(j)$ be the set of pieces of a job $j \in N$ completed strictly after $C_a^q(\pi)$. Note that, by the choice of piece q, the pieces in $K(j)$ are not early for each $j \in N$.

We will now construct a family of schedules $\pi(\varepsilon)$. In $\pi(\varepsilon)$, the starting and completion times of each piece k of each job j is changed by $\Delta_\varepsilon(S_j^k)$ and $\Delta_\varepsilon(C_j^k)$ with respect to schedule π. Note that $\Delta_\varepsilon(x)$ can be negative. The values Δ_ε are computed in the following way. For each each piece u of each job j, the change $\Delta_\varepsilon(S_j^u)$ of its starting time is equal to the change $\Delta_\varepsilon(C_i^v)$ of the completion time of the piece v of job i which immediately precedes u (or to zero, if u does not have a predecessor). The value $\Delta_\varepsilon(C_a^q)$ is set to ε, and this change is equaly compensated by the changes $\Delta_\varepsilon(C_a^k)$, $k \in K(a)$. Also, for each job j, the total change of the starting times of the pieces in $K(j)$ is distrubuted equaly among the changes of completion times of these pieces. Formally, we have:

$$\Delta_\varepsilon(C_j^u) = \begin{cases} \dfrac{\sum_{k\in K(j)} \Delta_\varepsilon(S_j^k) - \varepsilon}{|K(j)|}, & j = a \text{ and } u \in K(a), \\ \dfrac{\sum_{k\in K(j)} \Delta_\varepsilon(S_j^k)}{|K(j)|}, & j \neq a \text{ and } u \in K(j), \\ \varepsilon, & j = a \text{ and } u = q, \\ 0, & \text{otherwise.} \end{cases}$$

In order to avoid cycles in the computation, the values Δ_ε are computed in the non-decreasing order of starting and completion times in π mixed together. The way the values $\Delta_\varepsilon(S_j^k)$ and $\Delta_\varepsilon(C_j^k)$ are obtained, guarantees that the surface of each job remains the same. An example of changes Δ_ε for a fixed ε is illustrated in Figure 1.

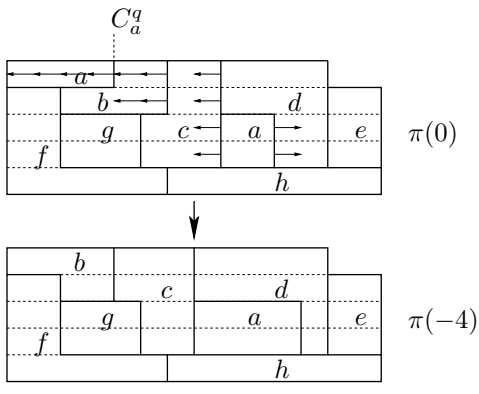

Fig. 1. An example of transformation ($\varepsilon = -4$)

Let $\Delta_\varepsilon(C_j) = C_j(\pi(\varepsilon)) - C_j(\pi)$, $j \in N$. Remember that if a piece u of job j is not early then $\Delta_\varepsilon(C_j^u) \neq 0$ only if $u \in K(j)$. Moreover, changes $\Delta_\varepsilon(C_j^u)$ and $\Delta_\varepsilon(C_j^v)$ of the completion times of two non-early pieces u and v of job j is the same for a fixed ε. Therefore, we have

$$\Delta_\varepsilon(C_j) = \begin{cases} \Delta_\varepsilon(C_j^u) \text{ for some } u \in K(j), & |K(j)| \geq 1, \\ 0, & |K(j)| = 0, \end{cases}$$

as long as, in $\pi(\varepsilon)$, all pieces which are early in π are not completed after pieces of the same job which are not early in π:

$$\begin{gathered} C_j^u(\pi) + \Delta_\varepsilon(C_j^u) \leq C_j^v(\pi) + \Delta_\varepsilon(C_j^v), \\ \forall j \in N, \forall u, v: \ u \notin K(j), \ v \in K(j). \end{gathered} \tag{2}$$

Let $\varepsilon_1 < 0$ be the smallest value of ε such that (2) is satisfied, and $\varepsilon_2 > 0$ be the largest value of ε such that (2) is satisfied. As $\Delta_\varepsilon(C_j^u)$ are linear functions of ε, the objective function $F(\pi(\varepsilon)) = F(\pi) + \sum_{j\in N} w_j \Delta_\varepsilon(C_j)$ is also a linear function of ε if (2) is satisfied.

Schedule $\pi(\varepsilon)$ is feasible as long as the following is true.

(1) For each piece, its starting time does not exceed its completion time:

$$\begin{gathered} S_j^u(\pi) + \Delta_\varepsilon(S_j^u) \leq C_j^u(\pi) + \Delta_\varepsilon(C_j^u), \\ \forall j \in N, \forall u. \end{gathered} \tag{3}$$

Let $\varepsilon_3 < 0$ be the smallest value of ε such that (3) is satisfied, and $\varepsilon_4 > 0$ be the largest value of ε such that (3) is satisfied.

(2) For each job j, number of its pieces executed simultaneously never exceeds δ_j. A sufficient condition for this is that, if a piece u precedes another piece v of the same job in π, u still precedes v in $\pi(\varepsilon)$:

$$\begin{gathered} C_j^u(\pi) + \Delta_\varepsilon(C_j^u) \leq S_j^v(\pi) + \Delta_\varepsilon(S_j^v), \\ \forall j \in N, \forall u, v: \ C_j^u(\pi) \leq S_j^v(\pi). \end{gathered} \tag{4}$$

Let $\varepsilon_5 \leq 0$ be the smallest value of ε such that (4) is satisfied, and $\varepsilon_6 \geq 0$ be the largest value of ε such that (4) is satisfied. If $\varepsilon_5 = 0$ or $\varepsilon_6 = 0$ then we have $C_j^u(\pi) = S_j^v(\pi)$ for a job j and its pieces u and v. Let piece u is executed on machine m_u and piece v is executed on machine m_v. By putting all pieces executed on m_u from time 0 to time $C_j^u(\pi)$ to machine m_v and vice versa, we decrease the number of pieces of job j by one, and the transformation is done. In the following, we suppose that $\varepsilon_5 < 0$ and $\varepsilon_6 > 0$.

Let $\underline{\varepsilon} = \max\{\varepsilon_2, \varepsilon_4, \varepsilon_6\} < 0$ and $\overline{\varepsilon} = \min\{\varepsilon_1, \varepsilon_3, \varepsilon_5\} > 0$. Now, for $\underline{\varepsilon} \leq \varepsilon \leq \overline{\varepsilon}$, schedule $\pi(\varepsilon)$ is feasible and $F(\pi(\varepsilon))$ is a linear function of ε. Therefore, as π, which is equivalent to $\pi(0)$, is an optimal schedule, $F(\pi(\varepsilon))$ should be a constant, and all schedules $\pi(\varepsilon)$, $\underline{\varepsilon} \leq \varepsilon \leq \overline{\varepsilon}$, are optimal.

Consider now optimal schedule $\pi(\underline{\varepsilon})$. There are three cases.

(1) $\underline{\varepsilon} = \varepsilon_2$. Then one of inequalities (2) is satisfied as an equality, and the corresponding piece u of job j, which was early in π, is not early in $\pi(\underline{\varepsilon})$.

(2) $\underline{\varepsilon} = \varepsilon_4$. Then one of inequalities (3) is satisfied as an equality, and the corresponding piece u of job j disappears in $\pi(\underline{\varepsilon})$.

(3) $\underline{\varepsilon} = \varepsilon_6$. Then one of inequalities (4) is satisfied as an equality, and, in $\pi(\underline{\varepsilon})$, we can exchange pieces

executed on machines m_u and m_v from time 0 to time $C_j^u(\pi(\underline{\varepsilon}))$ as we did above and decrease the number of pieces of job j by one.

4. CONCLUSION

In the paper, we have introduced the ascending property for the schedules of the problem considered. Then we have showed that the set of schedules satisfying this property is dominant. This result allows us to reduce significantly the search space for an algorithm which looks for an optimal schedule of the problem.

Note that the complexity status of a special case of our problem in which jobs do not have weights is open. Whether the set of schedules satisfying the ascending property is dominant was unknown even for this special case. Therefore, our result can also help to determine the complexity status of this important special case. Note that recently Hendel and Kubiak (2008) have shown that the problem $P2 \mid var,\ p_j(q) = p_j/q,\ \delta_j \mid \sum C_j$, i.e. the special case without weights and with two machines, is polynomially solvable.

ACKNOWLEDGEMENTS

The author is thankful to Yann Hendel who acquainted him with the problem.

REFERENCES

Blayo, E., Debreu, L., Mounié, G., and Trystram, D. (2000). Dynamic load balancing for adaptive mesh ocean circulation model. *Engineering Simulations*, 22, 8–24.

Drozdowski, M. (2004). Scheduling parallel tasks — algorithms and complexity. In J.T. Leung (ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman Hall, CRC Press.

Dutot, P.F., Mounié, G., and Trystram, D. (2004). Scheduling parallel tasks — approximation algorithms. In J.T. Leung (ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman Hall, CRC Press.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Hendel, Y. and Kubiak, W. (2008). Minimizing mean flow time for the two machines semi-malleable jobs scheduling problem. In F. Şerifoğlu and Ü. Bilge (eds.), *Proceedings of the 11th International Workshop on Project Management and Scheduling*, 136–140.

Ludwig, W. (1995). *Algorithms for Scheduling Malleable and Nonmalleable Parallel Tasks*. Ph.D. thesis, University of Wisconsin-Madison.

Serafini, P. (1996). Scheduling jobs on several machines with the jobs splitting property. *Operations Research*, 44(4), 617–628.

Turek, J., Wolf, J., and Yu, P. (1992). Approximate algorithms for scheduling parallelizable tasks. In *4th Annual ACM Symposium on Parallel Algorithms and Architectures*, 323–332.